

# Large-Scale Distributed Kalman Filtering via an Optimization Approach

Mathias Hudoba de Badyn\* Mehran Mesbahi\*

\*William E. Boeing Department of Aeronautics and Astronautics,  
University of Washington, Seattle, WA 98195 USA  
(e-mails: {hudomath,mesbahi}@uw.edu).

---

**Abstract:** Large-scale distributed systems such as sensor networks, often need to achieve filtering and consensus on an estimated parameter from high-dimensional measurements. Running a Kalman filter on every node in such a network is computationally intensive; in particular the matrix inversion in the Kalman gain update step is expensive. In this paper, we extend previous results in distributed Kalman filtering and large-scale machine learning to propose a gradient descent step for updating an estimate of the error covariance matrix; this is then embedded and analyzed in the context of distributed Kalman filtering. We provide properties of the resulting filters, in addition to a number of applications throughout the paper.

---

*Keywords:* Machine learning, fast Kalman algorithms, state estimation, gradient methods

---

## 1. INTRODUCTION

The Kalman filter is an algorithm that uses the known dynamics of a system to remove noise from measurements of that system. When considering large-scale dynamical systems, implementation of the standard or extended Kalman filters can be computationally difficult. In such cases, the Kalman filter requires the inversion of very large matrices at each timestep. This may cause the Kalman filter to run slower than the dynamical process it is trying to measure, or to severely reduce the temporal resolution of the measurements.

There are many systems for which measurements are taken by a network of sensors and are of high dimension; examples of such systems can be found in Khan and Moura (2008) and Kutz (2013). Previous methods for circumventing this problem include decomposing the dynamical system being measured into several subsystems and distributing the subsystems over the sensor network as in Khan and Moura (2008), or using Monte-Carlo methods for estimating the error covariance, such as in Furrer and Bengtsson (2007).

Previously, Sutton (1992) proposed to modify the Kalman filter error covariance update with a gradient descent method for the purpose of minimizing memory consumption, albeit for a specific instantiation of a SISO linear system. In this paper, we extend the gradient descent algorithm for estimating the Kalman filter error covariance to the general MIMO linear system as a proposed solution to the problem of running a Kalman filter on a high-dimensional system. We improve the gradient descent using Nesterov acceleration and adaptive learning rate methods. Lastly, we apply the methods above to distributed Kalman filtering on a sensor network.

---

\* This research was supported by the U.S. Army Research Laboratory and the U.S. Army Research Office under contract number W911NF-13-1-0340 and AFOSR grant FA9550-16-1-0022.

Distributed Kalman filtering seeks to estimate the state of a system by distributing the tasks of measuring the system and subsequently filtering the data to many agents, who then collectively assemble the state estimate. Such algorithms utilizing consensus to provide a global estimate were presented by Olfati-Saber (2005) and Olfati-Saber (2007), and later extended by Carli et al. (2008). Performance of distributed Kalman filters using graph-theoretic quantities were studied by Spanos et al. (2005).

The organization of the paper is as follows. In §2, we outline the mathematical notation and conventions we use. We discuss the gradient descent algorithm, Nesterov acceleration and adaptive learning rate methods in §3, and the distributed version of these algorithms with a relevant example in §4. The paper is summarized in §5.

## 2. MATHEMATICAL PRELIMINARIES

In this section, we lay out the mathematical notation used in this paper, and summarize the essential background on Kalman filtering.

Consider the *noisy discrete-time controlled linear system*

$$\begin{aligned}x_{k+1} &= Ax_k + Bu_k + \Upsilon_k w_k \\ \tilde{y}_k &= C_k x_k + v_k,\end{aligned}$$

where  $x_k \in \mathbb{R}^n$  denotes the *state vector*,  $u_k \in \mathbb{R}^m$  denotes the *control vector*,  $\tilde{y}_k \in \mathbb{R}^p$  is the *output vector* or *measurement*, and  $w_k, v_k$  are Gaussian white noise vectors of appropriate dimensions with covariances  $Q_k$  and  $R_k$  respectively. The subscript  $k$  refers to the timestep. The random vectors  $w_k$  and  $v_k$  represent system disturbances and sensor noise respectively, and the matrix  $\Upsilon_k$  describes how the disturbance propagates into the system. Since most matrix quantities will have a subscript  $k$  denoting the timestep, we will refer to the  $ij$ th entry of a matrix  $A_k$  with bracketed superscripts:  $A_k^{(ij)}$ . Similarly, the  $i$ th

entry of a vector  $a_k$  is denoted  $a_k^{(i)}$ , the  $i$ th column of a matrix  $A_k$  is denoted  $A_k^{[i]}$  and the  $i$ th row is denoted  $A_k^{\{i\}}$ .

A network is represented by a *graph*  $\mathcal{G} = (V, E)$  where  $V$  is a set of *nodes* representing agents in the network, and  $E$  is the set of edges  $\{i, j\}$  representing the connections between agents  $i$  and  $j$ . The *neighbourhood set*  $N_i$  of node  $i$  is the set of indices  $j$  such that  $\{i, j\}$  is an edge. We assume for simplicity that the graph is undirected, meaning that  $\{i, j\}$  and  $\{j, i\}$  represent the same edge.

The purpose of Kalman filtering is to provide an accurate estimate  $\hat{x}_k$  of the state  $x_k$  using the measurement  $\tilde{y}_k$  and the known information about the system and noise. The standard *discrete-time linear Kalman filter* yields the estimate  $\hat{x}_{k+1}$  and is given by

$$\begin{aligned}\hat{x}_{k+1} &= A(\hat{x}_k + K_k[\tilde{y}_k - C\hat{x}_k]) \\ K_k &= P_k C^T [C P_k C^T + R_k]^{-1} \\ P_{k+1} &= A(I - K_k C) P_k A^T + \Upsilon_k Q_k \Upsilon_k^T,\end{aligned}$$

where  $K_k$  is the *Kalman gain* and  $P_k$  is the *error covariance matrix*. The problem for high-dimensional systems is the inverse when computing  $K_k$ . In this paper, we assume that the sensor noise is uncorrelated, and so  $R_k$  is a diagonal positive-definite matrix. This leaves the term  $C P_k C^T$ ; if this term is diagonal, then the matrix inverse becomes a series of  $n$  scalar divisions. We propose to replace  $P_k$  with a diagonal estimate  $\hat{P}_k$ , and one can also assume that  $C$  is a matrix that measures individual states of the system without redundancy. Therefore, a system with  $1 \leq p \leq n$  outputs will have a measurement matrix of the form  $C = [\text{diag}(C^{(ii)}) I_p \mathbf{0}_{n-p}]$ . This assumption is common for networked systems where one measures the state of individual nodes of the network, see for example Chapman and Mesbahi (2015). In the next few sections, we will formalize these assertions in the context of our algorithm.

### 3. GENERAL KALMAN FILTERING

In this section, we extend the gradient descent estimate for the error covariance proposed by Sutton (1992). In §3.1, we discuss the linear Kalman filter, and then extend the error covariance estimate to an accelerated gradient descent in §3.2. The accelerated gradient descent benefits from a clever adaptive learning rate, which is discussed in §3.3. The final algorithm combining all these methods is summarized in Algorithm 4.

#### 3.1 Gradient Descent for the Error Covariance Update: Linear Case

Gradient descent is an iterative algorithm that seeks to find the local minimum of a function  $f$  by stepping in the direction of the largest gradient:

$$\beta_{k+1} = \beta_k - \mu \nabla f(\beta_k),$$

where  $\mu$  is the *learning rate*. For the Kalman filter, we seek to find a diagonal matrix estimate  $\hat{P}_k$  of  $P_k$ . We do this by assuming  $\hat{P}_k$  is of the form  $\hat{P}_k = \text{diag}(\hat{P}_k^{(ii)}) = \text{diag}(e^{\beta_k^{(i)}})$ , where  $\beta_k$  is a parameter undergoing gradient descent attempting to minimize the norm of the error  $\delta_k = \tilde{y}_k - C\hat{x}_k$ :

$$\beta_{k+1}^{(i)} = \beta_k^{(i)} - \frac{1}{2} \mu \frac{\partial(\delta_k^T \delta_k)}{\partial \beta_k^{(i)}}.$$

Computing the gradient of  $\delta_k^T \delta_k$  as outlined in Appendix A yields the following gradient descent equations for  $\hat{P}$ :

$$\begin{aligned}\beta_{k+1}^{(i)} &= \beta_k^{(i)} + \mu \delta_k^T C^{[i]} h_k^{(i)} \\ h_{k+1}^{(i)} &= h_k^{(i)} \left(1 - k_k^{(i)T} C^{[i]}\right)^+ + \left(k_k^{(i)} - k_k^{(i)} k_k^{(i)T} C^{[i]}\right)^T \delta_k \\ \hat{P}_k^+ &= \text{diag} \left( \exp \left( \beta_{k+1}^{(i)} \right) \right)\end{aligned}\quad (1)$$

The form of  $\hat{P}^+$  guarantees that it remains positive-definite, which is required to preserve convergence of the Kalman filter.

#### 3.2 Nesterov-Accelerated Methods

Nesterov acceleration is a method used to increase the convergence rate of gradient descent [Nesterov (1983)]. Although Nesterov-accelerated gradient descent converges in fewer timesteps, it does so by sacrificing monotonicity: the gradient descent trajectory will tend to oscillate as it converges towards the estimate.

In order to implement Nesterov acceleration, one must see that the quantity  $h_k^{(i)}$  is a function of  $K_{k-1}$ , which in turn is a function of  $\beta_k^{(i)}$ :

$$\begin{aligned}\beta_{k+1}^{(i)} &= \beta_k^{(i)} - \frac{1}{2} \mu \nabla f \left( \beta_k^{(i)} \right) \\ &= \beta_k^{(i)} + \mu \delta_k^T C^{[i]} h_k^{(i)} \left( K_{k-1} \left( \beta_k^{(i)} \right) \right).\end{aligned}$$

Hence, we can get the form of the Nesterov-accelerated gradient descent by introducing the quantity  $\alpha_k$ :

$$\begin{aligned}\beta_{k+1}^{(i)} &= \alpha_k^{(i)} - \frac{1}{2} \mu \nabla f \left( \alpha_k^{(i)} \right) \\ &= \alpha_k^{(i)} + \mu \delta_k^T C^{[i]} h_k^{(i)} \left( K_{k-1} \left( \alpha_k^{(i)} \right) \right)\end{aligned}$$

where  $\alpha_k^{(i)}$  is given by

$$\alpha_k^{(i)} = \beta_k^{(i)} + \frac{k-1}{k+2} \left( \beta_k^{(i)} - \beta_{k-1}^{(i)} \right).$$

Therefore, the Nesterov-accelerated estimate  $\hat{P}$  replaces the covariance update with the following equations:

$$\begin{aligned}\hat{P} &= \text{diag} \left( e^{\beta_{k+1}^{(i)}} \right), \quad k_k^{(i)} = \exp \left( \alpha_{k+1}^{(i)} \right) D_k^{-1} C^{[i]} \\ \beta_{k+1}^{(i)} &= \alpha_k^{(i)} + \mu \delta_k^T C^{[i]} h_k^{(i)} \left( K_{k-1} \left( \alpha_k^{(i)} \right) \right),\end{aligned}\quad (2)$$

where  $h_k$  is as in Equation (1), but using this new expression for  $k_k^{(i)}$  in Equation (2).

#### 3.3 Adaptive Learning Rate Methods

Using a constant learning rate  $\mu$  can lead to suboptimal performance, and therefore it is prudent to adapt  $\mu$  on-the-fly as the gradient descent is run. A well-known method for adapting the learning rate is given by Barzilai and Borwein Barzilai and Borwein (1988). Suppose we have the gradient descent with learning rate  $\mu$ :

$$\beta_{k+1}^{(i)} = \beta_k^{(i)} - \frac{1}{2} \mu \nabla f \left( \beta_k^{(i)} \right).$$

Let  $\Delta\beta = \beta_k - \beta_{k+1}$  and let  $\Delta g(\beta) = \nabla f(\beta_{k+1}) - \nabla f(\beta_k)$ . Then, the Barzilai and Borwein method selects the learning rate parameter using a secant line approximation:

$$\frac{1}{2}\mu_k = \arg \min_{\lambda} \|\Delta\beta - \lambda \Delta g(\beta)\|.$$

For accelerated gradient descent, the updated learning rate  $\mu$  is given by

$$\mu_k = 2 \frac{\Delta g(\beta)^T \Delta\beta}{\Delta g(\beta)^T \Delta g(\beta)}, \quad \Delta\beta = \beta_k - \alpha_{k-1}.$$

Combining the accelerated gradient descent estimate of the error covariance with the adaptive learning rate  $\mu_k$  yields the final algorithm as outlined in Algorithm 1.

---

**Algorithm 1** Discrete Kalman Filter with Accelerated Gradient Descent on  $P$  and Adaptive Learning Rate  $\mu$

---

- 1: *Initialize*
  - 2:  $\hat{x}(t_1) = \hat{x}_0$
  - 3:  $P_0 = \mathbb{E} \{ \tilde{x}(t_1) \tilde{x}(t_1)^T \}$
  - 4: *Loop:*
  - 5: **for**  $k = 1$  to  $k = t_f$  **do**
  - 6:   *Gain*
  - 7:    $K_k = \hat{P}_k^- C^T (C \hat{P}_k^- C^T + R_k)^{-1}$
  - 8:   *Accelerated Gradient Descent and Adaptive  $\mu_k$*
  - 9:    $\Delta\beta = \beta_k - \alpha_{k-1}$
  - 10:    $\Delta g(\beta) = \nabla f(\beta_{k+1}) - \nabla f(\beta_k)$
  - 11:    $D_k = R_k + C \hat{P}_k^- C^T$
  - 12:    $\mu_k = 2(\Delta g(\beta)^T \Delta\beta) / (\Delta g(\beta)^T \Delta g(\beta))$
  - 13:    $\alpha_k^{(i)} = \beta_k^{(i)} + \frac{k+1}{k+2} (\beta_k^{(i)} - \beta_{k-1}^{(i)})$
  - 14:    $\kappa_k^{(i)} = \exp(\alpha_k^{(i)}) D_k^{-1} C^{[i]}$
  - 15:    $\beta_{k+1}^{(i)} = \beta_k^{(i)} + \mu_k \delta_k^T C^{[i]} h_k^{(i)}$
  - 16:    $h_{k+1}^{(i)} = h_k^{(i)} (1 - \kappa_k^{(i)T} C^{[i]}) + (\kappa_k^{(i)} - \kappa_k^{(i)} \kappa_k^{(i)T} C^{[i]})^T \delta_k$
  - 17:   *Update*
  - 18:    $\hat{x}_k^+ = \hat{x}_k^- + K_k [\tilde{y}_k - C \hat{x}_k^-]$
  - 19:   *Propagate*
  - 20:    $\hat{x}_{k+1}^- = A_k \hat{x}_k^+ + B_k u_k$
  - 21:    $\hat{P}_{k+1}^- = \text{diag} \left( \exp \left( \beta_{k+1}^{(i)} \right) \right)$
- 

### 3.4 Filter Properties

In the standard Kalman filter,  $P_k$  is computed from an algebraic Ricatti equation. In order for this equation to have a positive-definite solution, a requirement for stable error dynamics, one must assume that  $(A^T, C^T)$  is stabilizable and  $(A, \sqrt{Q})$  is detectable (see Dorato et al. (1994)). Therefore, since our estimate of  $P$  is positive definite by construction and not computed by an algebraic Ricatti equation, we do not need to have these assumptions. The next theorem shows that all one needs for stability of the error dynamics is for the system matrix  $A$  to be stable.

*Theorem 1.* Suppose  $A$  is an asymptotically stable matrix, and so the eigenvalues of  $A$  satisfy  $|\lambda(A)| \leq 1$ . Then, the error dynamics

$$\tilde{x}_k = x_k - \hat{x}_k^-$$

are stable, where  $\hat{x}_k^-$  is the estimate computed according to Algorithm 1.

**Proof.** We can write the error dynamics as

$$\begin{aligned} \tilde{x}_{k+1} &= x_{k+1} - \hat{x}_{k+1}^- \\ &= (A - LC) \hat{x}_k^- + (w_k - Lv_k), \end{aligned}$$

where  $L$  is given by

$$L = A \hat{P} C^T (C \hat{P} C^T + R)^{-1}.$$

Hence, it suffices to show that  $(A - LC)$  has eigenvalues in the unit disc. We can write  $(A - LC)$  as

$$A - LC = A - A \hat{P} C^T (C \hat{P} C^T + R)^{-1} C = A(I - N),$$

where  $N = \hat{P} C^T (C \hat{P} C^T + R)^{-1} C$ . By the form of  $C$  and  $\hat{P}$ , if the system has  $p$  outputs, then  $N$  is a diagonal matrix with the form

$$N^{(ii)} = \begin{cases} \frac{e^{\beta_k^{(i)}} (C^{(ii)})^2}{e^{\beta_k^{(i)}} (C^{(ii)})^2 + R^{(ii)}} & \text{if } 1 \leq i \leq p \\ 0 & \text{if } p < i \leq n. \end{cases}$$

It is clear that  $0 \leq N^{(ii)} < 1$  since  $R$  is a positive definite diagonal matrix. Therefore,  $0 < (I - N)^{(ii)} \leq 1$ , and so it follows that

$$\begin{aligned} |\lambda(L)| &= |\lambda(A(I - N))| \\ &\leq |\lambda_{\max}(A)| |\lambda_{\max}(I - N)| \\ &\leq |\lambda_{\max}(A)|, \end{aligned}$$

and so by the asymptotic stability of  $A$ , the asymptotic stability of  $(A - LC)$  follows.

### 3.5 Example

Here, we show a simple numerical implementation of Algorithm 1 to see how the algorithm performs compared to the standard Kalman filter for different strengths of noise. Consider the linear system with matrices

$$A = \begin{bmatrix} 1.001 & 0.011 \\ -0.0301 & 0.98 \end{bmatrix}, B = \begin{bmatrix} 5E - 5 \\ 1E - 2 \end{bmatrix}, C = I$$

propagating with process noise of covariance  $Q = 0.001$  and measurement noise of covariance  $R = \alpha I$  for various values of  $\alpha$ . The first and second plots of Figure 1 show the error and first state of the system for  $R = 0.115I$  for the raw measurement data, and the filtered data using the accelerated gradient descent and the standard Kalman filter. The third plot shows the average steady state error computed on the time interval  $t \in [6.6, 10]$  for various values of  $\alpha$  indicating the covariance of the measurement error. Figure 1 shows that the filter in Algorithm 1 achieves similar performance as the standard Kalman filter.

## 4. DISTRIBUTED KALMAN FILTERING

Olfati-Saber (2007) describes several methods of Kalman filtering using a network of sensors, with each  $n$ th sensor using a unique sensor model  $\tilde{y}_{n,k} = C_{n,k} x_k + v_{n,k}$ , to arrive at a global measurement of a system. The context of their system model is on a graph  $\mathcal{G}$ , which is an abstract representation of the connection structure of sensors, which are represented by nodes. An edge in the graph denotes a communication link between sensors by which they can exchange information about their current state estimate. One such information exchange algorithm is called *consensus*, where each node in the graph changes

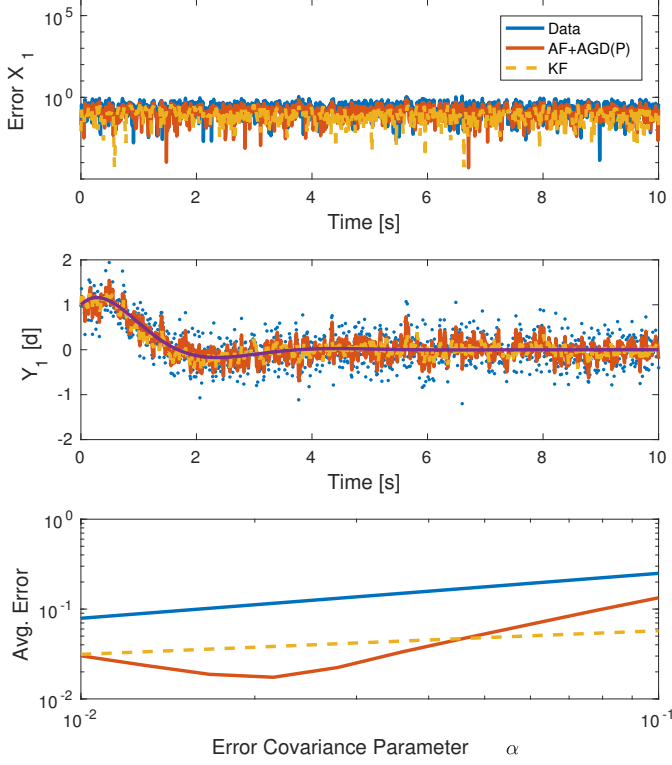


Fig. 1. Numerical test of Algorithm 1. Top: Error from true state from data, accelerated filter and Kalman filter. Middle:  $y_1 = x_1$  for data, accelerated filter and Kalman filter. Bottom: Average steady-state error.

its estimated value by continuously averaging with the estimates of its neighbours:

$$\dot{\hat{x}}_i = \sum_{j \in N_i} (\hat{x}_j - \hat{x}_i).$$

One particular algorithm in Olfati-Saber (2007) utilizes a *Kalman-consensus filter* in which every node of the network computes an estimate of the state of the system and then performs consensus with its neighbours on this estimate. This allows sensors that only see a portion of the system to collect a global state estimate over time.

For high-dimensional systems, implementing this filter is problematic again because each sensor has to invert a large matrix at each timestep in order to compute the Kalman filter gain. In Algorithm 2, we summarize the Kalman-consensus filter, where every node performs an accelerated gradient descent with adaptive learning rate, and shares its estimate with its neighbours. In the following section, we provide an example of such a high-dimensional system.

#### 4.1 Example

Consider a physical phenomenon, such as weather, propagating on a planar surface according to the partial differential equation

$$\dot{u} = \left( \alpha \frac{\partial^2 u}{\partial x^2} + \beta \frac{\partial^2 u}{\partial y^2} \right).$$

As studied in Khan and Moura (2008), and Kutz (2013), by discretizing the surface into an equally-spaced  $n \times n$  grid in the  $x$  and  $y$  directions and the temporal dimension, thus

#### Algorithm 2 Discrete Distributed Kalman-Consensus Filter $\mu$

- 1: *Initialize*
- 2:  $\hat{x}(t_1) = \hat{x}_0$
- 3:  $P_0 = \mathbb{E} \{ \hat{x}(t_1) \hat{x}(t_1)^T \}$
- 4:  $\epsilon = t_2 - t_1$
- 5: *Loop:*
- 6: **for**  $k = 1$  to  $k = t_f$  **do**
- 7:   **for** each node  $j \in V$  **do**
- 8:   *Aggregate Data*
- 9:    $S_j = |N_j|^{-1} \sum_{l \in N_j} C_l$
- 10:    $y_j = |N_j|^{-1} \sum_{l \in N_j} C_l^T \tilde{y}_l$
- 11:   *Accelerated Gradient Descent and Adaptive  $\mu_k$*
- 12:    $\Delta\beta = \beta_k - \alpha_{k-1}$
- 13:    $\Delta g(\beta) = \nabla f(\beta_{k+1}) - \nabla f(\beta_k)$
- 14:    $D_k = R_k + C \hat{P}_k C^T$
- 15:    $\mu_k = 2(\Delta g(\beta)^T \Delta\beta) / (\Delta g(\beta)^T \Delta g(\beta))$
- 16:    $\alpha_k^{(i)} = \beta_k^{(i)} + \frac{k+1}{k+2} (\beta_k^{(i)} - \beta_{k-1}^{(i)})$
- 17:    $\kappa_k^{(i)} = \exp(\alpha_k^{(i)}) D_k^{-1} C^{[i]}$
- 18:    $\beta_{k+1}^{(i)} = \beta_k^{(i)} + \mu_k \delta_k^T C_n^{[i]} h_k^{(i)}$
- 19:    $h_{k+1}^{(i)} = h_k^{(i)} (1 - \kappa_k^{(i)T} C_n^{[i]}) + (\kappa_k^{(i)} - \kappa_k^{(i)} \kappa_k^{(i)T} C_n^{[i]})^T \delta_k$
- 20:   *Update Kalman-Consensus Estimate*
- 21:    $\hat{x}_{j,k}^+ = \hat{x}_{j,k}^- + D_k^{-1} [y_j - S_j \hat{x}_{j,k}^- + \epsilon \sum_{l \in N_j} (\hat{x}_{l,k}^- - \hat{x}_{j,k}^-)]$
- 22:   *Propagate*
- 23:    $\hat{x}_{k+1}^+ = A_k \hat{x}_k^+ + B_k u_k$
- 24:    $\hat{P}_{k+1}^- = \text{diag}(\exp(\beta_{k+1}^{(i)}))$

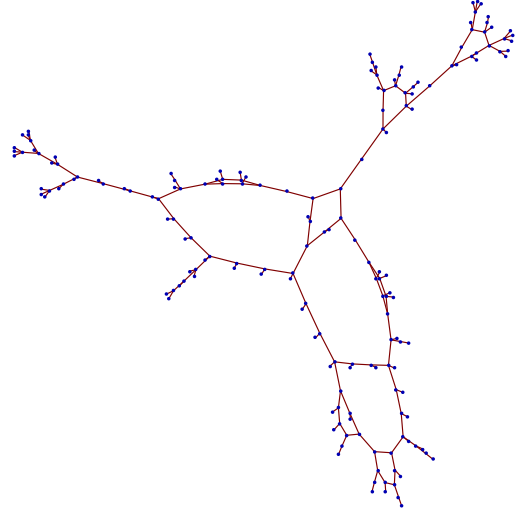


Fig. 2. Sensor network graph with 178 nodes and 186 edges

obtaining a grid of values  $u_{ij}^k$ , one can construct the linear system representation

$$\dot{\mathbf{u}} = \frac{2}{\Delta x^2} [I_n \otimes (\alpha D) + (\beta D) \otimes I_n] \mathbf{u}, \quad \mathbf{u} = \text{vec}(u_{ij}^k)$$

where  $D$  is a tridiagonal matrix with 1's on the sub and superdiagonal, and  $-2$ 's on the diagonal. The discrete-time state matrix  $A$  can be found using a matrix exponential, or a finite-order Taylor expansion. Periodic boundary conditions may be added by setting  $D_{1n} = D_{n1} = 1$ . Suppose that this phenomenon is being measured according to this discretization. We use a sensor model such that each node

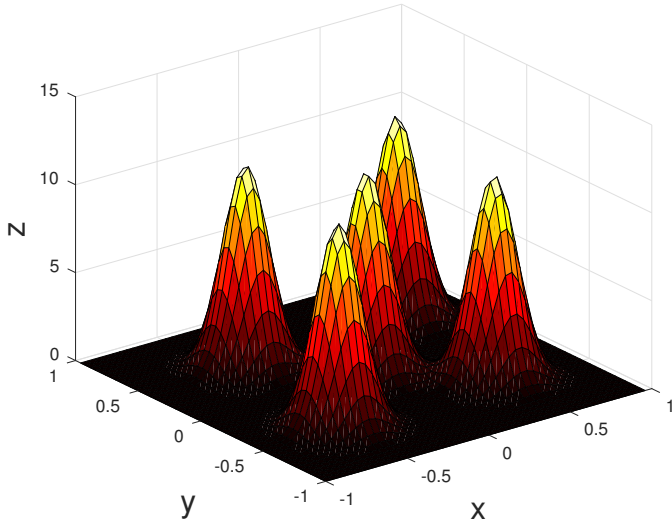


Fig. 3. True state of system at  $t = 1.2s$

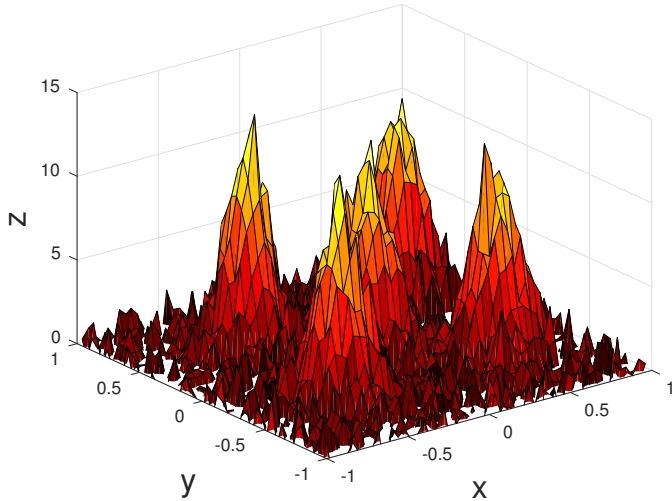


Fig. 4. Sample measurement  $\tilde{y}$  over the entire system at  $t = 1.2s$

collects data from an  $n_l \times n_l$  square portion of the grid  $u_{ij}^k$  so that each individual grid point is measured by at least one sensor. The sensor network by the graph in Figure 2 is used, with  $n_l = 4$  for a  $50 \times 50$  grid, and the sensor noise is given unit covariance. Five 2D Gaussian functions are used to set the initial conditions for the simulation, which was run over a time interval of 10 seconds. The true state of the system at 1.2s is shown in Figure 3, and an aggregate measurement of the entire system is shown in Figure 4. An estimate from a single sensor at this timestep is shown in Figure 5, and one can see that there is a substantial reduction of noise.

## 5. CONCLUSION

In this paper, we extended the results of Sutton (1992) to produce a gradient descent estimate of the error covariance in a standard state estimation algorithm for linear systems. This gradient descent method was improved with Nesterov acceleration and an adaptive learning rate algorithm.

Finally, the algorithm was extended to improve the Kalman-consensus filter on a sensor network observing

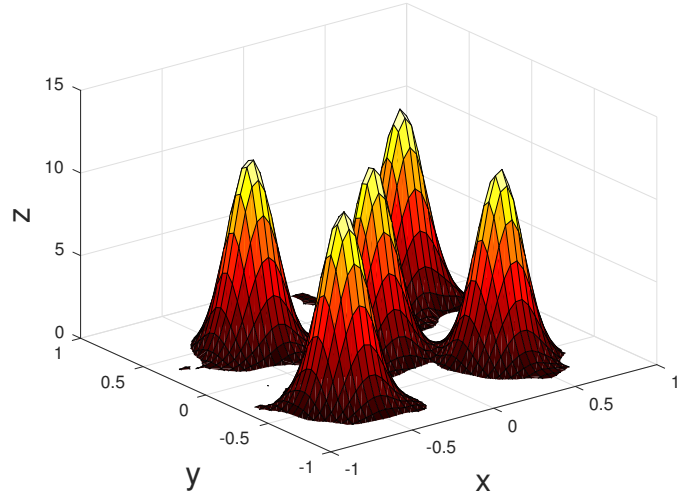


Fig. 5. Sample system estimate at  $t = 1.2s$  from a single sensor

a distributed high-dimensional process. The diagonal estimate of  $P$  removes the necessity of computing a large matrix inverse at each timestep at each sensor node, and allows for a computationally cheaper and therefore faster state estimation of the system over the entire network. The modified Kalman-consensus filter was implemented on a simple 2D diffusion model with 2500 grid points corresponding to the states of the system, and the algorithm was shown to remove a substantial amount of noise from the measurements, and to propagate the state estimate quickly across the sensor network.

## REFERENCES

- Barzilai, J. and Borwein, J.M. (1988). Two-point step size gradient methods.
- Carli, R., Chiuso, A., Schenato, L., and Zampieri, S. (2008). Distributed Kalman filtering based on consensus strategies. *IEEE Journal on Selected Areas in Communications*, 26(4), 622–633.
- Chapman, A. and Mesbahi, M. (2015). State controllability, output controllability and stabilizability of networks : A symmetry perspective. In *Proc. 54th IEEE Conference on Decision and Control*, 4776–4781. Osaka, Japan.
- Dorato, P., Cerone, V., and Abdallah, C. (1994). *Linear-Quadratic Control: An Introduction*. Simon & Schuster.
- Furrer, R. and Bengtsson, T. (2007). Estimation of high-dimensional prior and posterior covariance matrices in Kalman filter variants. *Journal of Multivariate Analysis*, 98(2), 227–255.
- Khan, U.A. and Moura, J.M.F. (2008). Distributing the Kalman filter for large-scale systems. *IEEE Transactions on Signal Processing*, 56(10), 4919–4935.
- Kutz, J.N. (2013). *Data-driven modeling & scientific computation: Methods for complex systems & big data*. Oxford University Press.
- Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate  $O(1/k^2)$ . In *Soviet Mathematics Doklady*, volume 27, 372–376.
- Olfati-Saber, R. (2005). Distributed Kalman filter with embedded consensus filters. In *Proceedings of the 44th IEEE Conference on Decision and Control and the European Control Conference*, 8179–8184. Seville, Spain.

Olfati-Saber, R. (2007). Distributed Kalman filtering for sensor networks. In *Proceedings of the 46th IEEE Conference on Decision and Control*, 5492–5498. New Orleans, USA.

Spanos, D.P., Olfati-Saber, R., and Murray, R.M. (2005). Approximate Distributed Kalman Filtering in Sensor Networks with Quantifiable Performance. In *Fourth International Symposium on Information Processing in Sensor Networks*, 133–139. Los Angeles, USA.

Sutton, R.S. (1992). Gain adaptation beats least squares? In *Proceedings of the Seventh Yale Workshop on Adaptive and Learning Systems*, 161–166.

## Appendix A. GRADIENT DESCENT FOR KALMAN FILTER COVARIANCE UPDATE

Sutton's Gradient Descent for updating the error covariance matrix approximation  $\hat{P} = \text{diag}(\hat{P}_k^{(i)}) = \text{diag}(\exp \beta_{k+1}^{(i)})$  uses gradient descent on  $\beta_{k+1}^{(i)}$ :

$$\beta_{k+1}^{(i)} = \beta_k^{(i)} - \frac{1}{2} \mu \frac{\partial (\delta_k^T \delta_k)}{\partial \beta_k^{(i)}},$$

where  $\delta_k = \tilde{y}_k - C\hat{x}_k^-$  is the error: To derive the exact form of the gradient descent on  $\beta_k^{(i)}$ , consider the following terms:

$$\frac{\partial (\delta_k^T \delta_k)}{\partial \hat{x}_k^{(i)}} = 2\delta_k^T \frac{\partial \delta_k}{\partial \hat{x}_k^{(i)}} = -2\delta_k^T C^{[i]}.$$

Next, we have

$$\frac{\partial \delta_k}{\partial \beta_k^{(i)}} = \frac{\partial}{\partial \beta_k^{(i)}} [\tilde{y}_k - C\hat{x}_k^-] = -\sum_{j=1}^n C^{[j]} \frac{\partial \hat{x}_k^{-(j)}}{\partial \beta_k^{(i)}}.$$

We assume that for  $i \neq j$ ,

$$C^{[j]} \frac{\partial \hat{x}_k^{-(j)}}{\partial \beta_k^{(i)}} \cong \mathbf{0} \implies \frac{\partial \delta_k}{\partial \beta_k^{(i)}} \cong -C^{[i]} \frac{\partial \hat{x}_k^{-(i)}}{\partial \beta_k^{(i)}}.$$

Next, define the following two quantities:

$$D_k := R_k + C\hat{P}_k C^T = R_k + \sum_{i=1}^n \exp(\beta_{k+1}^{(i)}) C^{[i]} C^{[i]T}$$

$$k_k^{(i)} := \exp(\beta_{k+1}^{(i)}) D_k^{-1} C^{[i]}.$$

Next, we compute the following derivatives:

$$\begin{aligned} \frac{\partial D_k}{\partial \beta^{(j)}} &= \sum_{i=1}^n \frac{\partial e^{\beta_{k+1}^{(i)}}}{\partial \beta^{(j)}} C^{[i]} C^{[i]T} = e^{\beta_{k+1}^{(j)}} C^{[j]} C^{[j]T} \\ \frac{\partial D_k^{-1}}{\partial \beta^{(j)}} &= -D_k^{-1} \frac{\partial D_k}{\partial \beta^{(j)}} D_k^{-1} = -k_k^{(i)} C^{[j]T} D_k^{-1} \\ \frac{\partial k_{k+1}^{(i)}}{\partial \beta^{(i)}} &= \left[ \frac{\partial e^{\beta_{k+1}^{(i)}}}{\partial \beta^{(i)}} D_k^{-1} + e^{\beta_{k+1}^{(i)}} \frac{\partial D_k^{-1}}{\partial \beta^{(i)}} \right] C^{[i]} \\ &= \beta_{k+1}^{(i)} D_k^{-1} C^{[i]} - e^{\beta_{k+1}^{(i)}} k_{k+1}^{(i)} C^{[i]T} D_k^{-1} C^{[i]} \\ &= k_{k+1}^{(i)} - k_{k+1}^{(i)} \left( e^{\beta_{k+1}^{(i)}} (D_k^{-1})^T C^{[i]} \right)^T C^{[i]} \\ &= k_{k+1}^{(i)} - k_{k+1}^{(i)} k_{k+1}^{(i)T} C^{[i]}. \end{aligned}$$

Now, we can write the gradient descent on  $\beta^{(i)}$ :

$$\begin{aligned} \beta_{k+1}^{(i)} &= \beta_k^{(i)} - \frac{1}{2} \mu \frac{\partial (\delta_k^T \delta_k)}{\partial \beta^{(i)}} \\ &= \beta_k^{(i)} - \frac{1}{2} \mu \sum_j \frac{\partial (\delta_k^T \delta_k)}{\partial \hat{x}_k^{-(j)}} \frac{\partial \hat{x}_k^{-(j)}}{\partial \beta^{(i)}} \\ &\cong \beta_k^{(i)} - \frac{1}{2} \mu \frac{\partial (\delta_k^T \delta_k)}{\partial \hat{x}_k^{-(i)}} \frac{\partial \hat{x}_k^{-(i)}}{\partial \beta^{(i)}} \\ &= \beta_k^{(i)} + \mu \delta_k^T C^{[i]} \frac{\partial \hat{x}_k^{-(i)}}{\partial \beta^{(i)}}. \end{aligned}$$

All that remains is to find an approximation to  $\frac{\partial \hat{x}_k^{-(i)}}{\partial \beta^{(i)}}$ . We denote this approximation with  $h_{k+1}^{(i)}$ :

$$\begin{aligned} h_{k+1}^{(i)} &\cong \frac{\partial \hat{x}_{k+1}^{-(i)}}{\partial \beta^{(i)}} = \frac{\partial}{\partial \beta^{(i)}} [\hat{x}_k^{(i)} + k_k^{(i)T} \delta_k] \\ &= \frac{\partial \hat{x}_k^{(i)}}{\partial \beta^{(i)}} + \frac{\partial k_k^{(i)T}}{\partial \beta^{(i)}} \delta_k + k_k^{(i)T} \frac{\partial \delta_k}{\partial \beta^{(i)}} \\ &= \frac{\partial \hat{x}_k^{(i)}}{\partial \beta^{(i)}} + \left( k_k^{(i)} - k_k^{(i)} k_k^{(i)T} C^{[i]} \right)^T \delta_k - k_k^{(i)T} \frac{\partial \hat{x}_k^{(i)}}{\partial \beta^{(i)}} C^{[i]} \\ &= h_k^{(i)} \left( 1 - k_k^{(i)T} C^{[i]} \right) + \left( k_k^{(i)} - k_k^{(i)} k_k^{(i)T} C^{[i]} \right)^T \delta_k. \end{aligned}$$

This yields the final gradient descent update:

$$\beta_{k+1}^{(i)} = \beta_k^{(i)} + \mu \delta_k^T C^{[i]} h_k^{(i)}.$$

In conclusion, the Kalman filter with gradient descent error covariance update is given by the equations

$$\hat{x}_{k+1} = \hat{x}_k + K_k [\tilde{y}_k - C\hat{x}_k]$$

$$K_k = \hat{P}_k C^T [C\hat{P}_k C^T + R_k]^{-1}$$

$$\hat{P} = \text{diag}(e^{\beta_{k+1}^{(i)}})$$

$$\beta_{k+1}^{(i)} = \beta_k^{(i)} + \mu \delta_k^T C^{[i]} h_k^{(i)}$$

$$h_{k+1}^{(i)} = h_k^{(i)} \left( 1 - k_k^{(i)T} C^{[i]} \right) + \left( k_k^{(i)} - k_k^{(i)} k_k^{(i)T} C^{[i]} \right)^T \delta_k.$$